PCT WORLD INTELLECTUAL PROPEI International Bui

WO 9607147A1

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30

A1

(11) International Publication Number: WO 96/07147

(43) International Publication Date: 7 March 1996 (07.03.96)

(21) International Application Number: PCT/AU95/00560

(22) International Filing Date: 30 August 1995 (30.08.95)

(30) Priority Data:

PM 7842	1 September 1994 (01.09.94)	AU
PM 9586	21 November 1994 (21.11.94)	AU

(71) Applicant *(for all designated States except US)*: DATACRAFT TECHNOLOGIES PTY. LTD. [AU/AU]; 252-254 Maroondah Highway, Mooroolbark, VIC 3138 (AU).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: HARVEY, Richard, Hans [AU/AU]; 4 Odette Court, Ringwood, VIC 3134 (AU).

(74) Agent: WATERMARK PATENT & TRADEMARK ATTORNEYS; 2nd floor, 290 Burwood Road, Hawthorn, VIC 3122 (AU).

(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT, UA, UG, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ, UG).

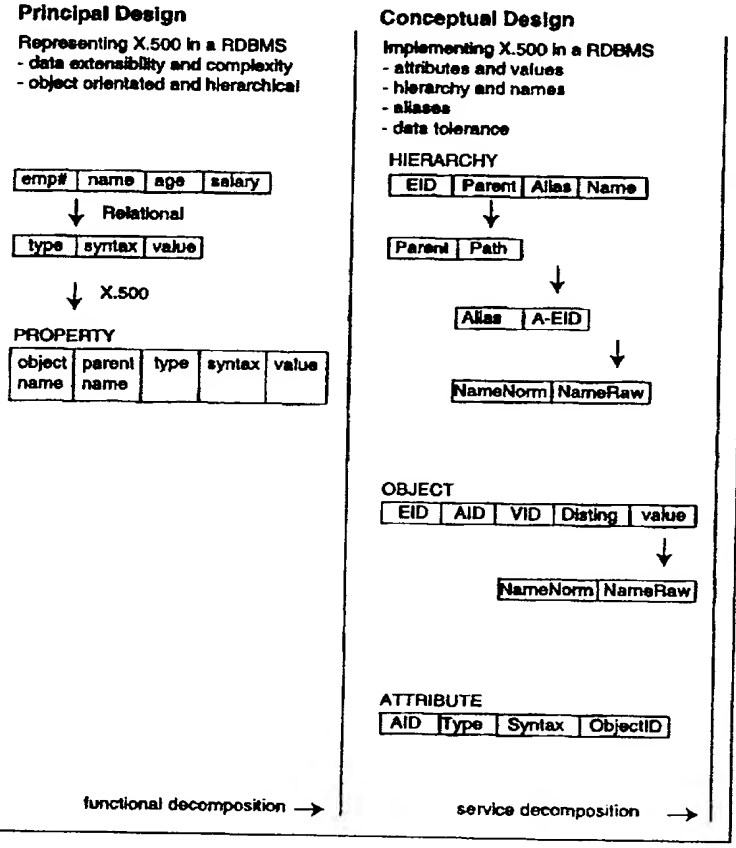
Published

With international search report.

(54) Title: X.500 SYSTEM AND METHODS

(57) Abstract

The present invention addresses the problem of implementing X.500 using an SQL product. The present application discloses an application of X.500 to a relational database, a database design and use of the database to perform X.500 services. Particularly, the disclosure relates to implementation using an RDBMS (Relational DataBase Management System). One invention disclosed resides around service modelling, the processing of arbitrary data using a fixed set of queries/services. Another invention resides in the implementation of a disk based model using relational queries to satisfy X.500 services and enables benefits of RDBMS to be exploited. Further, the invention provides an SQL based X.500 application that can perform at subsecond speed and is relatively unaffected by the size of database, DIT shape, type of data or complexity of service, including aliases.



X.500 SYSTEM AND METHODS

FIELD

The present invention is directed to application of X.500 to a relational database, a database design and use of the database to perform X.500 services. Particularly, the present invention is directed to implementation using a RDBMS (Relational Database Management System). The present invention is also directed to table structure and methods of operation of a database application.

PRIOR ART

X.500 is the International Standard for Electronic Directories [CCITT89]. These standards define the services, protocols and information model of a very flexible and general purpose directory. X.500 is applicable to information systems where the data is fairly static (e.g. telephone directory) but may need to be distributed (e.g. across organisations or countries), extensible (e.g. store names, addresses, job titles, devices etc.), object oriented (i.e. to enforce rules on the data) and/or accessed remotely.

Relational Database Management System

(RDBMS) provide facilities for applications to store and manipulate data. Amongst the many features that they offer are data integrity, consistency, concurrency, indexing mechanisms, query optimisation, recovery, roll-back, security. They also provide many tools for performance tuning, import/export, backup, auditing and application development.

RDBMS are the preferred choice of most large scale managers of data. They are readily available and known to be reliable and contain many useful management tools. There is a large base of RDBMS installations and therefore a large amount of existing expertise and investment in people and procedures to run these systems, and so data managers are looking to use this when acquiring new systems. Most relational database products support the industry standard SQL (Structured Query Language).

There has also been a move towards Object Oriented systems which provide data extensibility and the ability to handle arbitrarily complex data items. In addition, many corporations and government departments have large

information about each directory object, and is also incomplete in its implementation of X.500.

This approach has been discredited by a number of text books and knowledge in the art, such as "Object-Oriented Modeling and Design" by J. Rumbaugh, et al, 1991, ISBN 0-13-630054-5, in which at paragraph 17.3.8 it is clearly stated that "putting all entities in the one table is not a good approach to relational database design".

SUMMARY OF INVENTIONS

An object of the present inventions is to address the problem of implementing X.500 in a RDBMS which supports SQL or any other relational language.

The present application seeks to disclose a number of inventions related to the implementation of X.500 standards in a RDBMS which supports SQL or any other relational language.

The scope of the present invention is outlined in this specification, including the claims.

In this document, at the time of filing, SQL is the most popular relational language and although it is only one form of relational language, the intent of the present invention is to have application to any other form of relational language, not just SQL.

These inventions can be related to the following headings:

1. Principle Design
2. Conceptual Design
3. Conceptual Method(s)
4. Logical Design
5. Logical Method(s)
6. Physical Design
7. Example Implementation

The X.500 standard in no way dictates how the directory is to be implemented, only its capabilities and behaviour. One key to solving the implementation problem is the realization that X.500 defines a fixed set of services (e.g. Add, Modify, Search etc.) that can operate on arbitrary data.

- the ability to create X.500 databases of far greater size than previously possible, without compromising performance or robustness. The databases can be small or large (250,000, 1 million or more entries).
- an optimal table design minimises wastage of disk space.
- the ability to leverage off hundreds of man years of relational database developments and use "industrial strength" databases with proven reliability, integrity, security and tools for developing high performance applications.

Based on this unique approach, the following disclosure will detail a number of inventions in an order with reference to Figures 2A and 2B, which illustrates schematically an overview of the present X.500 system. The table and column, names, order of columns and numeric values disclosed are given on an arbitrary basis in the overview. The number of columns disclosed represent a preferred operable requirement. Additional columns do not alter the use of the table as herein contemplated.

1. PRINCIPLE DESIGN

The X.500 prior art attempts at implementation have been unable to overcome the relatively basic structural and operational differences between the X.500 requirements and functionality and SQL. The X.500 standard has a particular structure by nature, whereas SQL is designed to operate on relational structured tables.

For a typical relational database application, the nature of data is well known, i.e. tables will consist of a number of columns and each column contains data relating to a particular data type (see Table B1). The different data types that can be stored is limited to the columns of the table. The data types are also limited to the types supported by the database (e.g. string, numeric, money, date). The database may also store data of a form not understood by the database per se, but understood by the application e.g. binary data.

Name	Surname	Title	Phone
Chris Alana	MASTERS MORGAN	Sales Manager Sales Support	03 727-9456 03 727-9455 ....

Table B1: Employee Table

d. performance - if index is put on the type column, then each and every type is indexed.

2. CONCEPTUAL DESIGN

The prior art has had difficulty in implementing X.500 as it has not been structured for extensibility, object oriented and hierarchy which are requirements of X.500.

This is addressed, in one form, by functionally decomposing the 'property table' and thus resulting in what is called the Conceptual Design.

The conceptual design resides in providing at least one of:

1. Attribute table, where extensibility is addressed by allowing the definition of a new attribute type in this table by adding a row to the table;
2. Object table, which defines the attributes within each object; and/or
3. Hierarchy table, which defines the relationship between the objects.

In another invention, this problem is addressed by providing table structures in accordance with those disclosed in Figures 2A and 2B.

Yet further inventions reside in addressing problems of data tolerance by providing in the present X.500 system for the replacement of the 'value' column of the object table with value 'norm' and value 'raw' columns and/or replacing the RDN column in the hierarchy table with 'name norm' and 'name raw' columns.

Further, the difficulty in prior art of accommodating aliases is addressed in the present X.500 system by providing an 'alias' column in the hierarchy table. The 'alias' column is flagged to indicate that, that entry is an alias.

Further refinement may be provided by replacing the 'alias' column with alias and A-EID columns. The A-EID provides information about where the alias points.

Still further refinement may be provided by replacing the 'parent' column in the hierarchy table with 'parent' and 'path' columns.

The 'path' addresses the problem of implementing X.500 search, with aliases and subtrees. The 'path' has at least two unique properties: a) to

(speed, size) and secondary indexes may have large overheads (speed, size).

2. Enable data in tables to be clustered. Clustering occurs as a result of its primary key (storage structure) and thus data may be organised on disk around its key. E.g. for the 'search' table, surnames may be clustered together.

3. Management - smaller tables are easier to manage, e.g. faster to update indexes, collect statistics, audit, backup, etc.

4. Reduced I/O - speed improvements due to smaller rows, means more rows per page and thus operations perform less I/O's.

5. LOGICAL METHODS

A number of unique methods of interrogating the logical design tables are disclosed in the detailed description following.

In addition, one method resides in caching the attribute table. Thus, (with the exception of initial loading) no SQL statements are issued to the database. In the present X.500 system, conversions are performed in memory. This provides a substantial speed advantage.

Further, validation is performed in memory which avoids database roll-back. Roll-backs are time and system consuming.

Still further, for the arbitrary filter, a dynamic SQL equivalent is built. This enables arbitrary complexity in X.500 searches.

Also for search results, the present system utilizes set orientation queries of SQL to avoid 'row at a time' processing. Thus search results may be assembled in parallel in memory.

6. PHYSICAL DESIGN

New tables and new columns are introduced to overcome column width and key size restrictions and to achieve space optimisations.

The following text is a disclosure of embodiments of the inventions outlined:

1. PRINCIPLE DESIGN

With reference to Figure 2A, the principle design addresses the basic problem of representing the extensible, object oriented and hierarchical nature

will be NULL). Also, the data types are limited to the types supported by the database (e.g. string, numeric, money, date, etc.).

The solution is to treat the data types as generic. The present invention adopts the method of representing arbitrary attributes (e.g. XOM [X/OPEN Object Management] API [Application Programming Interface]) as a type, syntax, value combination (see Table 1.1c)

type	syntax	value
Name	String	Chris
Surname	String	MASTERS
Title	String	Sales Manager
Phone	Numeric	03 727-9456
Mobile	Numeric	018 042671

Table 1.1c - Representing arbitrary attributes

1.2 Object Oriented

X.500 defines objects (e.g. people, organizations, etc.) which may contain an arbitrary number of "attributes". Since many objects must appear in the table a mechanism is required to distinguish each object. An "object name" column is added to the table for this purpose (see Table 1.2a).

object name	type	syntax	value
Chris Masters	Name	String	Chris
Chris Masters	Surname	String	MASTERS
Chris Masters	Title	String	Sales Manager
Chris Masters	Phone	Numeric	03 727-9456
Chris Masters	Mobile	Numeric	018 042671
Alana Morgan	Name	String	Alana
Alana Morgan	Surname	String	MORGAN
Alana Morgan	Title	String	Sales Support
Alana Morgan	Phone	Numeric	03 727-9455

Table 1.2a - Representing objects with arbitrary values

The above method allows any number of attributes to be assigned (related) to an entry. These attributes could be of arbitrary complexity (e.g. a multi-line postal address could be handled). As the number of columns is fixed new attributes can be added to any object without having to redefine the

Note that the root of the tree has no parent. Thus, if both Chris and Alana work for Datacraft and Datacraft is a child of the root then we can say that Chris and Alana are children of Datacraft and that Datacraft is the parent of Chris and Alana.

2. CONCEPTUAL DESIGN

In Section 1 it was shown that a single Property Table could represent the extensible, object oriented and hierarchical nature of X.500 (see Table 2a).

object name	parent name	type	syntax	value

Table 2a - Property Table

With reference to Figure 2A in this section it will be shown that full X.500 functionality can be represented by using three tables as shown below (see Table 2b and Figure 2A).

Hierarchy Table

EID	Parent	Path	Alias	A_EID	NameNorm	NameRaw

Object Table

EID	AID	VID	Disting	ValueNorm	ValueRaw

Attribute Table

AID	Type	Syntax	ObjectId

Table 2b - Full Conceptual Design

The conceptual design addresses major problems with implementing full X.500 functionality in relational tables. As each major design issue is presented, examples are provided to illustrate the solution.

2.1 Functional Decomposition

The Property Table (Figure 2A) can be decomposed into separate tables that reflect the hierarchical, object oriented and extensible nature of X.500, preferably as follows;

- a Hierarchy Table which defines the structural relationship between objects.
- an Object Table which defines the attribute values within each object.
- an Attribute Table which defines the different attribute types.

These tables result from a process called functional decomposition.

2.2 X.500 Attributes

X.500 attributes have a protocol identifier which is transferred when any data is communicated between end systems. These identifiers are internationally defined and are called OBJECT IDENTIFIERS (e.g. 2.5.4.4 means a surname string). Thus an "ObjectId" column can be added to the Attribute table so that conversions between X.500 object identifiers and the internal attribute identifiers can be performed.

In addition, X.500 allows an attribute to have an arbitrary number of values (e.g. the mobile phone could be treated just as a second telephone number). Thus a "value identifier" or VID is introduced to identify values within an attribute in the Object Table.

Hierarchy Table

EID	Parent	Name
10	0	Datacraft
30	10	Chris Masters
31	10	Alana Morgan

Object Table

EID	AID	VID	Value
10	10	1	Datacraft
10	16	1	PO Box 123 CROYDON
30	3	1	Chris
30	4	1	MASTERS
30	12	1	Sales Manager
30	20	1	03 727-9456
30	20	2	018 042671
31	3	1	Alana
31	4	1	MORGAN
31	12	1	Sales Support
31	20	1	03 727-9455

Object Table

EID	AID	VID	Disting	Value
10	10	1	1	Datacraft
10	16	1	0	PO Box 123 CROYDON
30	3	1	1	Chris
30	4	1	1	MASTERS
30	12	1	0	Sales Manager
30	20	1	0	03 727-9456
30	20	2	0	018 042671
31	3	1	1	Alana
31	4	1	1	MORGAN
31	12	1	0	Sales Support
31	20	1	0	03 727-9455

Attribute Table

AID	Type	Syntax	ObjectId
3	Name	string	2.5.4.3
4	Surname	string	2.5.4.4
10	Organization	string	2.5.4.10
12	Title	string	2.5.4.12
16	Postal Address	address string	2.5.4.16
20	Phone	telephone string	2.5.4.20

Table 2.3 - Conceptual Design with X.500 attributes and names

2.4 X.500 Aliases

X.500 also has the concept of 'aliases'. An alias object effectively points to another entry and thus provides an alternate name for that entry. Thus an "alias" flag is added to the Hierarchy Table. When an alias is discovered during Navigation (i.e. the supplied DN contains an alias), then the alias value must be read from the Object Table. This alias DN must be resolved to where the alias points before Navigation of the original entry can continue.

An innovation is to use an "aliased EID" column or A_EID to store "where" the alias "points to". This removes the need to repeatedly navigate through an alias.

2.5 X.500 Data Tolerance

Every X.500 attribute has a (internationally defined) syntax. X.500 attribute syntaxes define how each attribute should be treated. In all string syntaxes (e.g. Printable, Numeric etc.) superfluous spaces should be ignored. In some syntaxes the case is not important (e.g. Case Ignore String and Case Ignore List) and so the names "Chris Masters", "Chris MASTERS" and " ChRis MaSTeRS " are considered identical.

In order to do comparisons (e.g. search for a particular value), the syntax rules can be applied to create a normalized form (e.g. "CHRIS MASTERS"). If this normalized form is stored in the database, then any variations in input form are effectively removed, and exact matching can be used (which is necessary when using SQL).

Both the normalized data and "raw" data are stored in the database. The "raw" data is necessary so that users can retrieve the data in exactly the same format as it was originally input. Thus the "Name" column in the Hierarchy Table becomes the "NameRaw" and a "NameNorm" column is added. Similarly, the "Value" column in the Object Table becomes the "ValueRaw" and a "ValueNorm" column is added.

Hierarchy Table

EID	Parent	Path	Alias	A_EID	NameNorm	NameRaw
10	0	10.	0	0	DATACRAFT	Datacraft
30	10	10.30.	0	0	CHRIS, MASTERS	Chris, MASTERS
31	10	10.31.	0	0	ALANA, MORGAN	Alana, MORGAN
35	10	10.35.	1	31	SUPPORT ENGINEER	Support Engineer

Object Table

EID	AID	VID	Disting	ValueNorm	ValueRaw

Attribute Table

AID	Type	Syntax	ObjectID

Table 3a Conceptual Table Design

The example hierarchy shown in Table 3b will be used to illustrate these services. Each name in the diagram represents an object entry in the database. The triangle represents an alias entry, and the dotted line represents the connection between the alias entry and the object that it points to. The numbers next to each entry are the entry EID's.

In the example, entry "1" has an RDN with a value of "Datacraft", entry "11" has an RDN with a value of "Sales", entry "20" has an RDN with a value of "Network Products" and entry "31" has an RDN with a value of "Alana Morgan". The DN of entry "31" is made up of a sequence of RDN's, namely, ""Datacraft", "Sales", "Network Products", "Alana Morgan".

The alias entry "Datacraft/Networks" points to the entry "Datacraft", "Sales", "Network Products". When navigating to this entry the navigate process would find the alias entry, then find the DN of the object pointed to by the alias and then navigate from the root to the object entry returning an EID of "20" and a path of "1.11.20.".

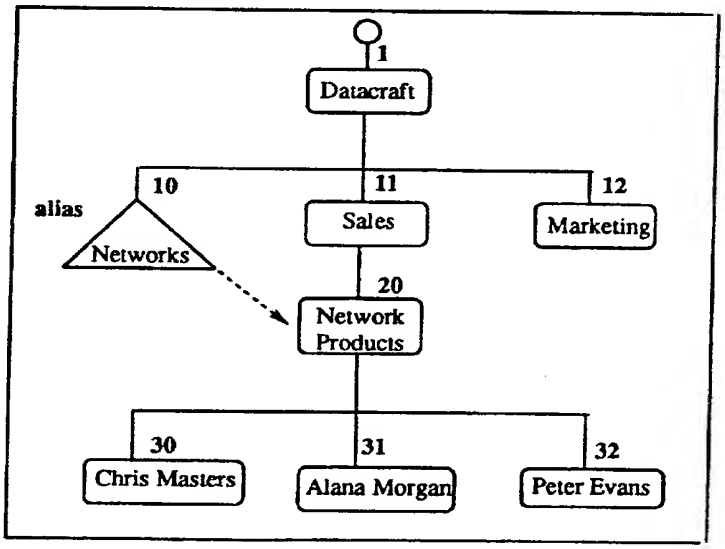


Table 3b: Entry Tree

Distinguished Names

For the entry shown in the sample Object Table (Table 3d) two of the attributes, commonName and surname, are *distinguished values* (or naming values) which combine to form the RDN for the entry. This RDN is stored in the Hierarchy Table.

Multi-valued Attributes

In X.500, it is permissible for an attribute to be multi-valued. The VID column is used to distinguish between values for an attribute. In the sample Object Table, the telephoneNumber attribute is multi-valued.

3.1 Mapping Services to SQL

3.1.1 Attribute Types and Values

Any data supplied by an X.500 service is supplied as a list of ObjectId's and their associated values. These must be converted into AID's (using the Attribute table) and normalized values (using the Object table) for use by the X.500 application. The database returns data as AID's and Raw Values, which must then be converted into ObjectId's and their associated values in the X.500 result.

3.1.2 Navigation

Each X.500 service supplies a Distinguished Name which is converted into an EID for use by the X.500 application. When the application processes a service it returns one or more EID's. These EID's can then be translated back into Distinguished Names in the X.500 result.

All X.500 services rely on navigating the directory tree. To navigate to a particular entry, the following procedure is performed:

- Given the DN for the entry, locate the entry in the hierarchy table which has an RDN equal to the first RDN in the DN.
- Store the EID.
- Recursively, locate the entry which has an RDN equal to the next RDN in the DN and a parent equal to the stored EID.

Example

- Compare the telephone Number "03 727 9256" with the entry "Datacraft/Sales/Network Products/Chris Masters".

Navigate to the entry and then;

```
select VALUERAW from OBJECT
where EID = 30
and AID = 20
and VALUENORM = "03 727 9456"
```

If a value is selected then return "matched" else return "not matched".

3.1.5 List

Navigate to the required entry. Store the EID. In the Hierarchy Table, return the RDN's for all rows with a parent matching the stored EID.

Example

- List from the entry "Datacraft/Sales".

Navigate to the entry and then;

```
select NAMERAW from HIERARCHY
where PARENT = 11
```

3.1.6 Add Entry

Navigate to the required parent entry. Store the EID of the parent. Add a new EID to the Hierarchy table and add rows to the Object table for each value in the new entry.

Example

- Add a new entry under the entry "Datacraft/Sales/Network Products".

Navigate to the entry and then;

```
insert into OBJECT
(EID, AID, VID, DISTING, VALUENORM, VALUERAW)
values (33, 3, 1, 1, EDWIN MAHER, Edwin Maher)
and
insert into HIERARCHY
(EID, PARENT, PATH, ALIAS, A-EID, NAMENORM, NAMERAW)
values (33, 20, 1.11.20.33.,0 ,0 , EDWIN MAHER, Edwin Maher)
```

3.1.9 Modify RDN

Navigate to the required entry. Check that the new name (RDN) does not exist in the current level of the subtree (i.e. that the new DN is distinct). Store the EID. Modify the entry in the Hierarchy and Object tables.

Example

- Modify the RDN of the entry "Datacraft/Sales/Network Products/Chris Masters" to "Christine Masters".

  Navigate to the entry and then;

```
select EID from HIERARCHY
where PARENT = 20
and VALUENORM = "CHRISTINE MASTERS"
```

If no entries are returned then the new RDN may be inserted. First set the old RDN to be a non-distinguished value.

```
update OBJECT
set DISTING = 0
where EID =  30 and VALUENORM = "CHRIS"
```

and

```
update HIERARCHY
set NAMENORM = "CHRISTINE MASTERS" and
set NAMERAW = "Christine Masters"
where EID = 30
```

and

```
insert into OBJECT
(EID, AID, VID, DISTING, VALUENORM, VALUERAW)
values (30, 3, 1, 1, "CHRISTINE", "Christine")
```

3.2 Search Strategy

The most powerful and useful X.500 service is the search service. The search service allows an arbitrary complex filter to be applied over a portion of the Directory Information Tree (the search area).

- A filter is a combination of one or more filter items connected by the operators AND, OR and NOT. For example; surname = "MASTERS" AND title = "SALES MANAGER"

the subtree and then apply the filter to all entries that have a path which is prefixed by the path of the base object (for example; to search for all entries under "Sales" we perform a search where PATH LIKE 1.11.%).

Base Object Search:

Navigate to the base object. Store the EID. In the Object Table, read nominated values from rows which match the stored EID where a filter criteria is satisfied, eg, telephone prefix = "727".

Example

- Search from the base object "Datacraft/Sales/Network Products" for an entry with surname = "MORGAN", using a "base-object-only" search. Navigate to the base object and then;

```
select AID, VALUERAW from OBJECT
where EID = 20 and AID = 4
    and NAMENORM = "MORGAN"
```

One Level Search:

Navigate to the base object. Store the EID. Return the list of EID's which have a parent EID matching the stored EID (in Hierarchy table) and have values which satisfy the filter criteria (OBJECT table). In the Object Table, read nominated values for the returned EID's.

Example

- Search from the base object "Datacraft/Sales/Network Products" for an entry with surname = "MORGAN", using a "one-level-only" search. Navigate to the base object and then;

```
select H.EID from HIERARCHY H, OBJECT O
where PARENT = 20 and AID = 4 and NAMENORM = "MORGAN"
    and H.EID = O.EID
```

then place the EID's returned into an EIDLIST and

```
select AID, VALUERAW from OBJECT
where EID in [EIDLIST]
```

Subtree Search:

Navigate to the base object. Store the EID. Return the list of all EID's with a path like that of the base object (Hierarchy table) and have values which

the filter is applied. They also restrict the search area in that any dereferenced aliases are excluded from the search area.

Aliases and OneLevel Search

If aliases are being dereferenced as part of a one level search and an alias entry is found then the alias must be resolved (using the Object table or the A_EID). The aliased object is then added to the search area to which the filter is applied. In a oneLevel search where aliases are found the search area will consist of non-alias entries directly subordinate to the base object and all dereferenced aliases.

Aliases and Subtree Search

If aliases are being dereferenced as part of a whole subtree search and an alias entry is found then the alias must be resolved (using the Object table or the A_EID) and this EID must then be treated as another base object, unless it is part of an already processed sub tree.

When dereferencing aliases during a search the "Path" column can be used to find alias entries within a subtree join. If an alias entry is found that points outside of the current subtree then the subtree pointed to by the alias can also be searched for aliases. One property of the hierarchical tree structure is that each subtree is uniquely represented by a unique base object (i.e. subtrees do not overlap). When performing a subtree search we build up a list of base objects which define unique subtrees. If no aliases are found then the list will contain only one base object. If an alias is found that points outside of the subtree being processed then we add the aliased object to the list of base objects (unless one or more of the base objects are subordinate to the aliased object in which case the subordinate base object(s) are replaced by the aliased object). The search area will therefore consist of non-alias entries that have a path prefixed by the path of one of the base objects.

4. LOGICAL DESIGN

Whilst the Conceptual Design (see Table 4a) is sufficient to implement the X.500 functionality, further performance improvements can be made.

Hierarchy Table

EID	Parent	Path	Alias	A_EID	NameNorm	NameRaw

ATTR

AID	SYNTAX	DESC	OBJECTID

Table 4b - Logical Design

4.1 Service Decomposition

The practical reality for most RDBMS's is that big tables with many columns do not perform as well as smaller tables with fewer columns. The major reasons are to do with indexing options, I/O performance and table management (see Sections 4.5 and 4.6). This is why prior art relational design techniques aim to focus primary information into separate tables and derive secondary information via table joins (i.e. normalization and fragmentation techniques).

One innovation in achieving X.500 performance is to decompose the tables around primary service relationships and derive secondary services via joins. This process is called service decomposition. The following considerations are made:

(1) Columns that have strong relationships are preferred to be kept together (to avoid unnecessary joins);

(2) If the number of significant rows in a given column is independent of the other related columns, then that given column is a candidate for a separate table.

(3) If a column is only used for locating information (input) or only used for returning results (output) then it is a candidate for its own table.

(4) If a column is used as a key for more than one service then it is preferred to be a primary key and therefore in its own table (each table can have only one primary key).

(5) Keys are preferred to be unique or at least strong (non-repetitious).

NameNorm = "DATACRAFT") is the basis for Navigation and update checking (checking for the existence of an object before an Add or ModifyRdn). Acting on entries that have a given parent is used during List or OneLevel Search. Thus the DIT (Directory Information Tree) table has information required for Navigation, but allows its PARENT column to be used by other services.

EID	PARENT	ALIAS	RDN

Table 4.2b - DIT Table

An object is differentiated from its siblings via its Relative Distinguished Name (RDN). RDN's are returned for a List (in conjunction with a given Parent) or as part of a full Distinguished Name (Read, Search). Thus the NAME table has information required for returning names (the raw RDN).

EID	RAW

Table 4.2c - NAME Table

An object's absolute position in the hierarchy is necessary for building DN's (from which the raw RDN's are retrieved) and for expanding subtrees during Search. Thus the TREE table has information about an entry's Path (the sequence of EID's down from the root).

EID	PATH

Table 4.2d - TREE Table

Alias information is cached so that every time an alias is encountered during Navigate it does not have to be repeatedly resolved. Thus the ALIAS table only contains entries that are aliases. It is also used during OneLevel Search (in conjunction with the DIT Parent column) and Subtree Search (in conjunction with the Path column) to determine if there are any aliases in the search area.

EID	A_EID

Table 4.2e - ALIAS Table

4.3 Object Table Decomposition

The Object table contains the following columns:

EID	AID	VID	Disting	ValueNorm	ValueRaw

Table 4.3a - Object Table

The Object Table essentially contains information for finding a particular value (e.g. AID = surname, ValueNorm = "HARVEY") and for retrieving values

fully indexed access. A composite index on both type and value may be required.

One innovation of the table decomposition in the previous sections is to maximise the use of primary indexes across tables. This reduces the number of secondary indexes (i.e. they become primary indexes on their own table). Following is a list of the indexes for each of the six tables used in the logical design.

Table	Primary Key	Secondary Index
DIT	PARENT, RDN	EID
NAME	EID	
TREE	PATH	EID
SEARCH	AID, NORM	EID, AID, VID
ENTRY	EID, AID, VID	
ATTR	(cached)	

Table 4.5 - Table Indexes for the Logical Design

The table design means that many queries can be handled without joins, giving substantial performance improvement.

The joins that are considered necessary are listed below:

- List - for returning the RAW-RDNs under a given object (DIT joined with NAME).
- Search / Subtree - for finding EIDs that match a filter over a whole subtree (where the base object is not the root) (TREE joined with SEARCH).
- Search / OneLevel - for finding EIDs that match a filter one-level under the base object (DIT joined with SEARCH).
- Search / Aliases / Subtree - for finding all the aliases in a subtree (TREE joined with ALIAS).
- Search / Aliases / OneLevel - for finding all the aliases under a given object (DIT joined with ALIAS).

Note that the above joins are first level joins (i.e. between only two tables). It is preferable not to use higher order joins.

4.6 Input/Output Performance

An innovation of decomposing tables around services, which increases the number of tables, is that the new tables are much smaller than the

5. LOGICAL METHODS

This section describes methods of interrogating the Logical Design tables, with reference to Figure 2B.

Throughout this section, each X.500 method is defined and illustrated with an example. Table 5a displays a small hierarchy tree which includes an alias reference. The corresponding Table contents are shown in Table 5b.

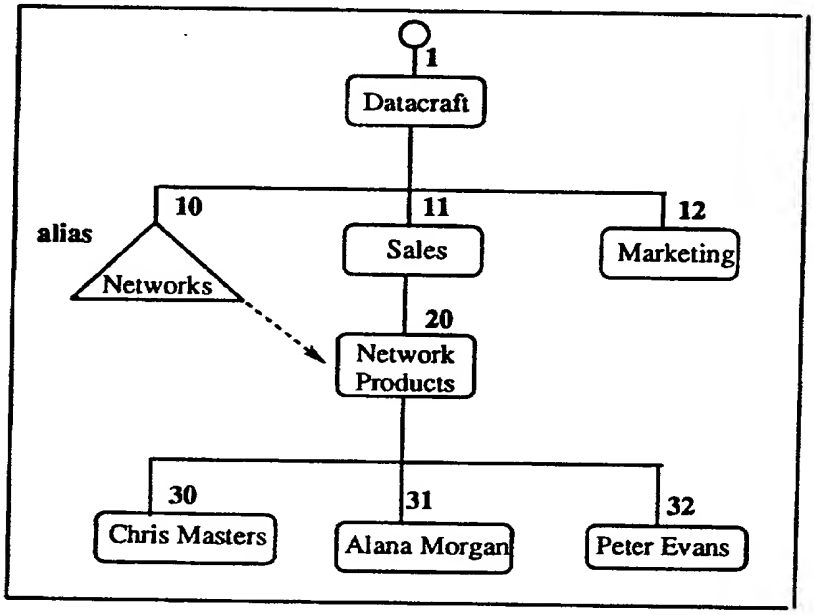


Table 5a - Simple Hierarchy Tree

DIT

EID	PARENT	ALIAS	RDN
1	0	0	DATACRAFT
10	1	1	NETWORKS
11	1	0	SALES
12	1	0	MARKETING
20	11	0	NETWORK PRODUCTS
30	20	0	CHRIS MASTERS
31	20	0	ALANA MORGAN
32	20	0	PETER EVANS

ATTRIBUTE

AID	SYX	DESC	OBJECTID
0	objectIdentifierSyntax	objectClass	2.5.4.0
1	distinguishedNameSyntax	aliasedObjectName	2.5.4.1
3	caseIgnoreStringSyntax	commonName	2.5.4.3
4	caseIgnoreStringSyntax	surname	2.5.4.4
7	caseIgnoreStringSyntax	localityName	2.5.4.7
8	caseIgnoreStringSyntax	stateOrProvinceName	2.5.4.8
9	caseIgnoreStringSyntax	streetAddress	2.5.4.9
10	caseIgnoreStringSyntax	organizationName	2.5.4.10
11	caseIgnoreStringSyntax	organizationalUnitName	2.5.4.11
12	caseIgnoreStringSyntax	title	2.5.4.12
13	caseIgnoreStringSyntax	description	2.5.4.13
16	PostalAddress	postalAddress	2.5.4.16
17	caseIgnoreStringSyntax	postalCode	2.5.4.17
18	caseIgnoreStringSyntax	postOfficeBox	2.5.4.18
20	telephoneNumberSyntax	telephoneNumber	2.5.4.20

ENTRY

EID	AID	VID	RAW
1	0	0	[2.5.6.4]
1	10	0	[Datacraft]
1	16	0	[266-268 Maroondah Highway]
1	17	0	[3138]
10	0	0	[2.5.6.1]
10	1	0	[Datacraft / Sales / Network Products]
11	0	0	[2.5.6.5]
11	11	0	[Sales]
11	13	0	[Sales Department]
12	0	0	[2.5.6.5]
12	11	0	[Marketing]
12	13	0	[Marketing Department]
20	0	0	[2.5.6.5]
20	11	0	[Network Products]
20	13	0	[Network Products Section]
30	0	0	[2.5.6.7]
30	3	0	[Chris]
30	4	0	[Masters]
30	12	0	[Sales Manager]
30	20	0	[(03) 727-9456]
30	20	1	[(018) - 042 671]
31	0	0	[2.5.6.7]
31	3	0	[Alana]
31	4	0	[Morgan]
31	12	0	[Sales Support]
31	20	0	[(03) 727-9455]
32	0	0	[2.5.6.7]
32	3	0	[Peter]
32	4	0	[Evans]
32	12	0	[Salesperson]
32	20	0	[(03) 727-9454]

Table 5b: Example Tables

NOTE: [....] indicates a binary encoding of the exact data entry value.

At this stage we discover that this entry is an alias. The Alias Table is checked to see if the EID of the alias has been cached. If this is the first time an attempt has been made to resolve this alias then the A_EID column in the Alias Table will be zero. For the purpose of discussion it will be assumed that this is the first time.

To resolve the alias, the DN of the aliased object must be determined. This is stored in the "aliasedObjectName" attribute of the alias entry. The aliasedObjectName has an AID = 1 (from the ATTR table) and so the DN is obtained from the Entry Table (RAW value) where EID = 10 and AID = 1.

In this example, the DN of the alias is "Datacraft / Sales / Network Products". This DN is resolved completely using the normal tree walking technique. The value of EID is 20.

At this stage, navigation continues for the unresolved RDN's in the original DN, namely "PETER EVANS". The last step required is then:

- Scan the DIT table for a row containing PARENT = 20 and RDN = "PETER EVANS".

Once an alias has been resolved it can be added (cached) in the Alias Table. This table contains a reference, A_EID, to the aliased object. In the above example, an entry in the Alias Table with an EID of 10 would have an A_EID of 20. Once an alias has been cached a tree walk is no longer necessary to resolve the alias.

Directory Paths

When objects are added to the DIT table, a corresponding row is added to another table called the Tree Table. This table stores the list of the EID's which identify a "Path" to the object.

Distinguished Names

Most services require the distinguished name to be returned in the Service Result. Using the directory path from the Tree Table, a DN can be constructed from the RAW RDN values stored in the Name Table.

X.500 definition

Argument	**Description**
Name	A Distinguished Name
EntryInformationSelection	The attributes and values to be returned (ie EIS)
Common Arguments	
Result	**Description**
Entry Information	The DN plus any attributes and values returned
Common Results	

Method

- Perform a tree walk using the DIT table, resolving aliases if necessary. Obtain the base EID.
- Using PATH from the Tree Table and the RAW RDN's from the Name Table, build a DN.
- If EIS specifies no attributes or values, just return the DN.
- If EIS specifies ALL types and values, return the RAW values from the Entry Table for the matching EID.
- If EIS specifies selected types and values, obtain the AID's from the Attribute Table and then return selected types and/or values for the matching EID .

Example:

Read the entry "Datacraft / Sales / Network Products / Peter Evans".

EIS is set to: attribute Types = allAttributes, InfoTypes = attributeTypesAndValues.

Using the DIT table perform a Tree Walk traversing EID's 1, 11, 20 and 32 for the normalised RDN's DATACRAFT, SALES, NETWORK PRODUCTS, PETER EVANS. The EID of the selected object is 32.

Extract the PATH from the Tree Table for EID = 32. The PATH is 1.11.20.32.

Build aDN from the RAW values in the Name Table for EID's 1, 11, 20, 32.

Using the Entry Table and the Attribute Table, for each matching EID;

- If an alias is dereferenced, return the DN of the selected object, using the path from the Tree Table and the RAW RDN's from the Name Table.

Example

Compare the DN "Datacraft / Sales / Network Products / Peter Evans" with a purported AttributeValueAssertion of "title = [Salesperson]".

Obtain the EID for the given DN using a TreeWalk. The EID of the selected object is 32.

Using the Attribute table, obtain the AID for "title", ie AID = 12.

Using the Search Table locate rows with EID = 32 and AID = 12 and test for "NORM = SALESPERSON".

Return TRUE or FALSE depending on the outcome of this test. In this instance the result would be TRUE.

Since no aliases were dereferenced, the DN of the entry is not returned.

5.4 List Service

A list operation is used to obtain a list of immediate subordinates of an explicitly identified entry.

X.500 definition

Argument	**Description**
Name	A Distinguished Name
Common Arguments	
Result	**Description**
DistinguishedName	The DN of the selected object (returned if an alias is dereferenced)
subordinates	A list of RDN's for the subordinate entries (aliases, indicated by an alias flag, are not dereferenced)
partialOutcomeQualifier	An indication that an incomplete result was returned, eg, a time limit or size limit restriction.
Common Results	

Method

- Perform a tree walk using the DIT table, resolving aliases if necessary. Obtain the EID of the base object.

X.500 definition

Argument	Description
baseObject	The Distinguished Name of the baseObject
subset	baseObject, oneLevel or wholeSubtree
filter	search conditions
searchAliases	a flag to indicate whether aliases among subordinates of the base object should be dereferenced during the search.
selection	EIS as for READ. The attributes and values to be returned.
Common Arguments	
Result	**Description**
DistinguishedName	The DN of the selected object (returned if an alias is dereferenced)
entries	Attributes & values (as defined in selection) for the entries which satisfy the filter.
partialOutcomeQualifier	An indication that an incomplete result was returned, eg, a time limit or size limit restriction.
Common Results	

The search procedures for each search scope are outlined as follows:

Base Object

- Perform a tree walk using the DIT table, resolving aliases if necessary. Obtain the EID of the base object.
- Apply the filter to attributes and values in the Search Table with the EID of the selected object.
- If the filter condition is matched, return the Entry Information from the Entry Table.
- If an alias is dereferenced, return the DN using the Tree Table to extract the PATH and the Name Table to build the DN.

Example

Perform a search on the baseObject "Datacraft / Sales" with:

- Scope set to WholeSubtree
- a Filter of "surname, substring initial = M". (Look for all surnames beginning with "M")
- SearchAliases set to TRUE.
- EIS set to attributeTypes = allAttributes, InfoTypes = attributeTypesAndValues.

Method

Obtain the EID for the base object DN using a TreeWalk. The EID of the base object is "11".

From the Tree Table, obtain the PATH for EID = 11, ie, "1.11".

Check for any aliases among entries that have a path beginning with "1.11.". There are no aliases in this case.

Obtain the AID for the attribute "surname" in the Attribute Table, ie, 4.

Apply the filter and scope simultaneously. i.e. Using the Search Table, obtain a list of EID's from the target list where AID = 4 and the value begins with "M" joined with the Tree Table who's PATH is LIKE '1.11.%'. The matching EID's are 30 and 31.

Using the Entry Table and the Attribute Table, for each matching EID:

- return the OBJECTID's from the Attribute Table and the ASN.1 encoded RAW values from the Entry Table

ie,

2.5.4.0,	[2.5.6.7],
2.5.4.3,	[Chris],
2.5.4.4	[Masters]
2.5.4.9	[Sales Manager]
2.5.4.20	[(03) 727-9456]
2.5.4.20	[(018) - 042 671]
2.5.4.0	[2.5.6.7]
2.5.4.3	[Alana]
2.5.4.4	[Morgan]
2.5.4.9	[Sales Support]
2.5.4.20	[(03) 727-9454]

DIT

EID	PARENT	ALIAS	RDN
33	11	0	MARY DELAHUNTY

NAME

EID	RAW
33	[Mary Delahunty]

TREE

EID	PATH
33	1.12.21.

SEARCH

EID	AID	VID	DISTING	NORM
33	0	0	0	2.5.6.7
33	3	0	1	DELAHUNTY
33	4	0	1	MARY
33	12	0	0	MARKETING MANAGER
33	20	0	0	03 727 9523

ENTRY

EID	AID	VID	RAW
33	0	0	[2.5.6.7]
33	3	0	[Delahunty]
33	4	0	[Mary]
33	12	0	[Marketing Manager]
33	20	0	[(03) 727-9523]

5.7 Remove Entry Service

A RemoveEntry operation is used to remove a leaf entry (either an object entry or an alias entry) from the Directory Information Tree.

X.500 definition

Argument	Description
object	The Distinguished Name of the entry to be modified
changes	A list of modifications
Common Arguments	
Result	**Description**
NULL	NULL

Method

- Perform a tree walk using the DIT table. Obtain the EID of the selected object.

For the selected object, perform one or more of the following actions: Add Value, Delete Value, Add Attribute, Delete Attribute

The operations required for each action are as follows:

Add Value

- If the attribute exists, add the value to the Entry Table and the Search Table. Checks are: If the attribute is single valued test for an existing value; if the attribute is multi-valued check for a duplicate value.

Delete Value

- For the Entry Table and the Search Table, if the value exists, delete it. A Distinguished Value cannot be deleted.

Add Attribute

- If the attribute does not exist, add the Attribute Values to the Entry Table and the Search Table.

Delete Attribute

- For the Entry Table and the Search Table, if the attribute exists, delete it. Delete all values with AID = attr and EID = base object. Naming attributes cannot be deleted.

Example

Modify the Entry "Datacraft / Sales / Network Products / Chris Masters" with the following changes:

Method

- Perform a tree walk using the DIT table. Obtain the EID and Parent EID of the base object.
- Using the DIT table, check for equivalent entries and return error if one is found. An equivalent entry has RDN = new RDN and PARENT = Parent EID.
- Using the Name Table, replace the old RDN with the new RDN.
- Using the DIT Table, replace the old RDN with the new RDN.
- Using the Entry Table, insert the new value.
- Using the Search Table, locate value = old RDN and set DISTING to 0. Insert the new value.

If *deleteOldRDN* is set to TRUE the procedures following the Tree Walk are as follows:

- Using the DIT table, check for a sibling with the same name and an EID not equal to the base EID
- Using the Name Table, replace the old RDN with the new RDN.
- Using the DIT Table, replace the old RDN with the new RDN.
- Using the Entry Table, delete the old value(s) and insert the new value(s).
- Using the Search Table, delete the old value(s) and insert the new value(s).

Example

Modify the RDN of "Datacraft / Sales / Network Products / Chris Masters". The new RDN is "Christine Masters".

deleteOldRDN is set to FALSE.

The changes to the Tables will be as follows:

DIT

EID	PARENT	ALIAS	RDN
21	11	0	CHRISTINE MASTERS

NAME

EID	RAW
21	[Christine Masters]

Time Limit indicates the maximum elapsed time, in seconds, within which the service shall be provided. Size Limit (only applicable to List and Search) indicates the maximum number of objects to be returned. If either limit is reached an error is reported. For a limit reached on a List or a Search, the result is an arbitrary selection of the accumulated results.

Abandon

Operations that interrogate the Directory, ie Read, Compare, List and Search, may be abandoned using the Abandon operation if the user is no longer interested in the results.

Aliases & Search

If an alias is encountered in a search and that alias points to a separate branch of the directory tree, then dereferencing of the alias requires:

- Navigation from the root entry to the referenced entry
- Searching of all items subordinate to the referenced entry

In the example shown in Table 5.10, if a WholeSubtree Search was performed on a base object of "Telco / Corporate / Data Services" the entries "Mervyn Purvis" and the alias "*Strategic*" would be searched. Strategic, however, points to a different branch of the tree which requires searching of the entry "Strategic" and all of its subordinates, ie, "Alan Bond", "Rex Hunt", "Wayne Carey" and "John Longmire".

Optimise Query Handling

As the format of most services is known, many instances of these services can be resolved using static SQL statements. More complex services, such as searches with complex filters, can be resolved using dynamic SQL. This enables arbitrarily complex searches to be performed.

Parallel Queries

Also when processing search results the present system utilizes set orientation queries of SQL to avoid 'row at a time' processing. Thus search results may be assembled in parallel in memory.

Data Storage

The tables that store raw data store the data in ASN.1 format. This provides an efficient means of transfering data into or out of the database.

Database Techniques

Complex services can be further improved by using the query optimiser, which provides a mechanism for reducing the time spent in resolving the query. The use of a relational database also provides an efficient use of memory and enables large databases to be constructed without the need for large amounts of memory being available. Many other X.500 applications cache the entire database in memory to achieve performance. This method consumes large amounts of memory and is not scalable.

6. PHYSICAL DESIGN

The physical design results from a process called physical transformation of the logical design. The physical design represents a preferred realization or embodiment of the logical design. Figure 2B and the tables below show one form of the physical design. New columns and tables are highlighted by double borders.

6.1 Efficiency

INFO Table

This table holds the highest EID value that has been used in the database. The inclusion of the INFO table enables the next EID to be obtained without any calculation of the maximum EID being performed by the database. This provides improved efficiency in adding entries to the database. More importantly the inclusion of the INFO table removes contention problems which may occur when multiple DSA's are adding entries at the same time.

Shadow Keys

Three tables have had shadow keys added. These are:

a) The NORMKEY column in the SEARCH table.

b) The RDNKEY column in the DIT table.

c) The LEV1, LEV2, LEV3 and LEV4 columns in the TREE table.

Each of these shadow key columns is a shortened version of a larger column. They have been added to shorten the indexes on each table. This gives improved performance for any queries that use the indexes and it also improves disk space usage as small indexes take up less space than large indexes.

The shadow keys in the PATH table utilise the structured nature of the PATH. By being a composite key then exact matching can be used in the SQL instead of the "LIKE" operator.

e.g. WHERE LEV1 = 1 AND LEV2 = 10 AND ...

instead of WHERE PATH LIKE '1.10.%'.

If each of the LEV columns has their own index, then a sub-tree search needs to only use the base object. e.g. LEV2=10, since all objects under entry 10 will have LEV2=10.

SENTRY Table

Some types of attribute values do not need to be normalised e.g. integer, boolean, date. Instead of storing them twice (SEARCH.NORM and ENTRY.RAW) they can be stored just once in a hybrid table called the SENTRY table. This reduces table sizes and increases storage efficiency at the cost of having to search two tables and retrieve from two tables.

also means that each value will have only one entry in the ENTRY table and that the ENTRY and SEARCH tables maintain their one-to-one correlation. Secondly the use of a BLOB table enables the application to make use of any database support for Binary Large Objects. (e.g. 64K Binary Columns).

6.3 Functional Extensibility

FLAGS Columns

FLAGS column(s) are preferred to be added. These column(s) have been added to provide extensibility to the design. Specific values can be added to the flags as new functionality is required, without changing the table structure.

Note:

a) In the SEARCH table, the DISTING field may be absorbed into the FLAGS field.

b) In the DIT table, the ALIAS field may be absorbed into the FLAGS field.

The FLAGS column(s) may also provide a "summary" function for each of the tables. This means that the nature of an entry can be determined to some extent by checking the value of the FLAGS field. For example, a flag can be set, in the DIT table, when an entry is a leaf. Checking this flag is much simpler than checking for children of the entry.

The FLAGS column can also be used to store security information, whether an alias points inside its parents sub-tree, whether a value is a BLOB, etc.

7. EXAMPLE IMPLEMENTATION

The following provides an example of system performance and capabilities. It is to be understood that the present inventions should not be limited to the following disclosure.

7.1 Overall system benefits

The present invention is considered to provide enhanced performance over prior art implementations. Performance can be appraised in many ways, including:

Service			Database Size (number of entries)					
Operation	Qualifier	Detail	1K	10K	20K	50K	100K	200K
BIND	anonymous		0.00	0.00	0.00	0.00	0.00	0.00
LIST	level 1	4 items	0.05	0.05	0.05	0.05	0.05	0.05
	level 3	4 items	0.06	0.06	0.06	0.06	0.06	0.06
	level 4	100 items	0.22	0.23	0.23	0.24	0.23	0.24
READ	level 4	1 item, all info	0.07	0.07	0.07	0.07	0.07	0.08
	level 4 (via alias)	1 item, all info	0.07	0.07	0.07	0.07	0.07	0.07
SEARCH	1 level, equality	100 entries, 1 item	0.12	0.12	0.12	0.12	0.13	0.13
	1 level, initial	100 entries, 1 item	0.13	0.14	0.15	0.15	0.15	0.14
	1 level, any	100 entries, 1 item	0.30	0.35	0.33	0.32	0.36	0.29
	1 level, final	100 entries, 1 item	0.24	0.35	0.31	0.30	0.35	0.28
	subtree, equality	1K, 1 item, level 1	0.11	0.11	0.11	0.11	0.11	0.11
		10K, 1 item, level 1	xxx	xxx	0.12	0.12	0.12	0.12
		20K, 1 item, level 1	xxx	xxx	xxx	0.12	0.13	0.12
		50K, 1 item, level 1	xxx	xxx	xxx	xxx	0.13	0.13
		100K, 1 item, level 1	xxx	xxx	xxx	xxx	xxx	0.12
	subtree, initial	1K, 1 item, level 1	0.13	0.12	0.12	0.12	0.12	0.11
		10K, 1 item, level 1	xxx	xxx	0.11	0.12	0.12	0.12
		20K, 1 item, level 1	xxx	xxx	xxx	0.13	0.12	0.12
		50K, 1 item, level 1	xxx	xxx	xxx	xxx	0.13	0.12
		100K, 1 item, level 1	xxx	xxx	xxx	xxx	xxx	0.11
	full, complex OR	all entries, 1 item	0.09	0.09	0.09	0.09	0.09	0.09
	full, complex AND	all entries, 1 item	0.11	0.11	0.11	0.11	0.11	0.11
	full, complex OR/AND	all entries, 1 item	0.26	0.28	0.29	0.28	0.29	0.26
	full, complex AND/OR	all entries, 1 item	0.12	0.12	0.13	0.14	0.13	0.12
	full, complex AND/AND	all entries, 1 item	0.16	0.15	0.16	0.17	0.18	0.18
	full, complex AND/AND/AND	all entries, 1 item	0.18	0.18	0.18	0.19	0.20	0.26
	full, equality	all entries, 1 item	0.08	0.08	0.08	0.08	0.08	0.08
	full, no filter, all-info	all entries, 10 items	0.30	0.74	0.43	0.59	0.49	0.67
	full, no filter, all-info	all entries, 100 items	1.36	1.84	1.50	1.79	1.82	1.86
	full, initial	all entries, 1 item	0.08	0.08	0.08	0.08	0.08	0.08
ADD	level 5	100 sisters	0.22	0.19	0.22	0.20	0.19	0.19
MODIFY	level 5	100 sisters	0.09	0.11	0.11	0.11	0.11	0.11
RENAME	level 5	100 sisters	0.15	0.16	0.15	0.16	0.16	0.15
DELETE	level 5	100 sisters	0.17	0.16	0.17	0.17	0.17	0.19
UNBIND			0.00	0.00	0.00	0.00	0.00	0.00

Table 7A

6. All complex searches, in test, were performed in under one second. However, there may be some obscure searches (e.g containing combinations of NOT) which may not perform as well.

Because this is a disk based system (rather than a memory based system) performance is essentially only dependent on the number of entries actually returned. It is relatively independent of the search complexity, the depth of the hierarchy, the number of attributes per entry or the types of attributes used in the query. In a "live" application of the system it may be possible to improve on the achieved test results by tuning the caching parameters, and by having a greater diversity of attributes.

10. A method of searching an area in a database, the method comprising the steps of applying a filter and a scope.

11. A method of searching an area in a database which includes aliases, the method comprising the steps of:

expanding the search area by resolving aliases until a set of search areas is found; and

applying a filter and a scope to the set of search areas.

12. A method as claimed in claim 10 or 11, wherein the database uses X.500.

13. A method as claimed in claim 10 ,11 or 12, wherein evaluating the filter and the scope is performed by single pass resolution.

14. A method of implementing X.500 to a relational database, the method comprising:

processing arbitrary data using a fixed set of queries / services.

15. A method as claimed in claim 14, where the database is a relational database with SQL.

16. A method as claimed in claim 15, where the database has 250,000 or more entries.

17. A method of implementing X.500 on a relational database, the method comprising the step of applying functional decomposition to a property table.

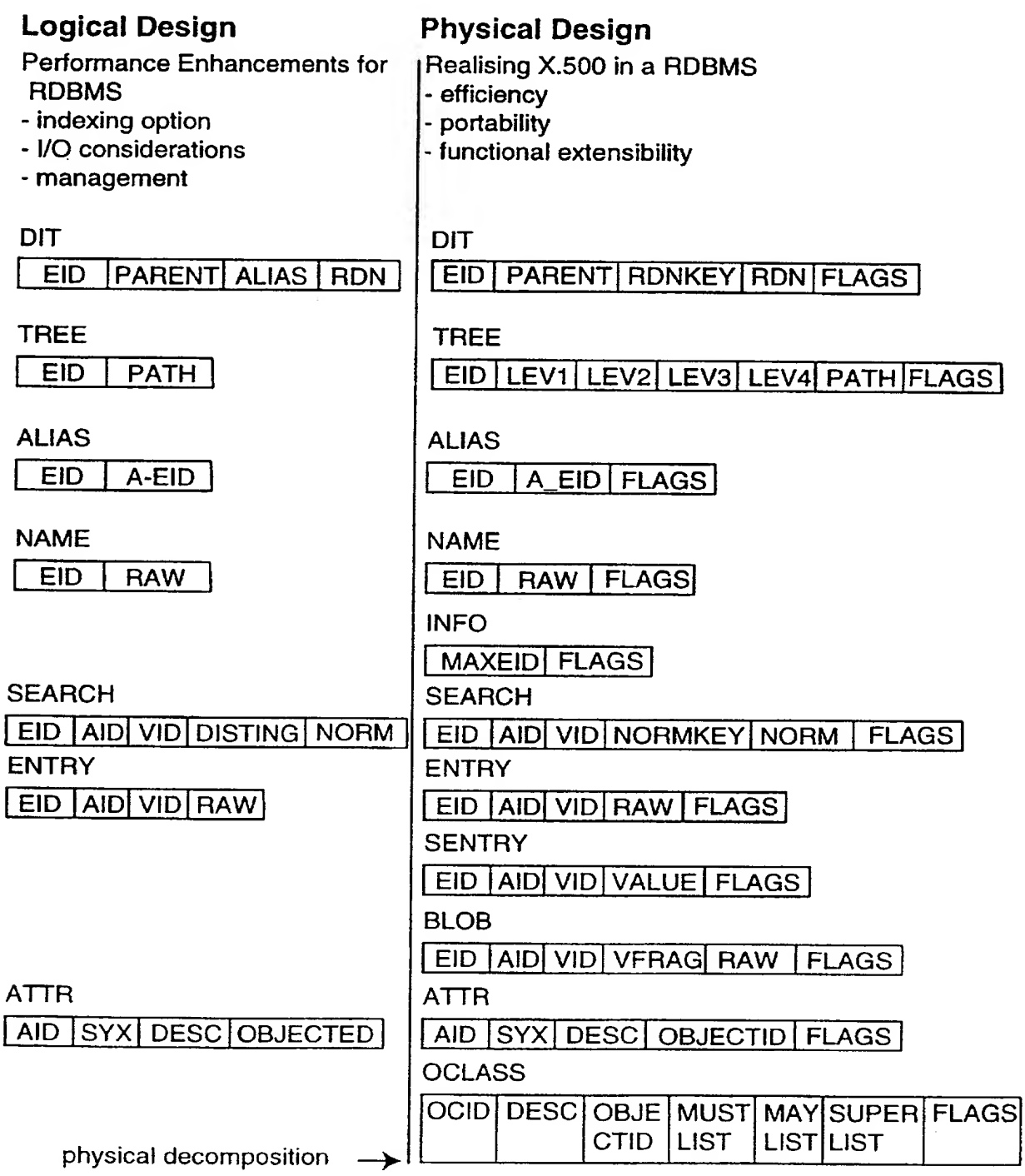
18. A method as claimed in claim 17, further comprising the step of service decomposition.

Fig 1.

OSI Directory Products

3Com	1	0	1
Alcatel TITN	1	1	1
Boldon James	1	0	1
BT	1	1	1
Brunel University	1	0	1
Control Data Systems	1	1	1
Cray Research	1	1	1
Data Connection	3	1	2
Digital	2	1	2
GPT	1	1	1
Hewlett Packard	1	1	1
IBM	5	5	5
ICL	5	4	4
ISODE Consortium	1	1	1
MARBEN Produit	2	2	2
Motorola Comp Group	1	1	0
NCR	1	1	1
NEXOR	3	1	2
OISware	1	1	1
Olivetti	1	1	1
Retix	1	1	1
Siemens Nixdorf	2	2	2
Software Kenetics	1	1	1
Stratus Computer	2	2	2
Tandem Computers	2	2	2
Unisys	5	5	5
Wang	1	1	1
Wollongong Group	2	2	2

Fig 2B.


Logical Design
Performance Enhancements for RDBMS
- indexing option
- I/O considerations
- management
Physical Design
Realising X.500 in a RDBMS
- efficiency
- portability
- functional extensibility
DIT
EID PARENT ALIAS RDN
TREE
EID PATH
ALIAS
EID A-EID
NAME
EID RAW
SEARCH
EID AID VID DISTING NORM
ENTRY
EID AID VID RAW
ATTR
AID SYX DESC OBJECTED
DIT
EID PARENT RDNKEY RDN FLAGS
TREE
EID LEV1 LEV2 LEV3 LEV4 PATH FLAGS
ALIAS
EID A_EID FLAGS
NAME
EID RAW FLAGS
INFO
MAXEID FLAGS
SEARCH
EID AID VID NORMKEY NORM FLAGS
ENTRY
EID AID VID RAW FLAGS
SENTRY
EID AID VID VALUE FLAGS
BLOB
EID AID VID VFRAG RAW FLAGS
ATTR
AID SYX DESC OBJECTID FLAGS
OCLASS
OCID DESC OBJECTID MUST LIST MAY LIST SUPER LIST FLAGS
physical decomposition

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/ AU 95/00560

A. CLASSIFICATION OF SUBJECT MATTER

Int Cl[6]: G06F 17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC G06F 17/30, 15/40, 15/403

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
AU : IPC as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
DERWENT
JAPIO & INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	The Proceedings of IFIP WG6.6 International Symposium (ISBN: 0444 889 167) Francois Perruchond, Cuno Lanz, and Bernard Plattner "A Relational Data Base Design for an X.500 Directory System Agent". - pages 405-418.	1-3,9
Y	"Object-Oriented Modeling and Design" by J. Rumbaugh, et al, 1991, ISBN 0-13-630054-5, pages 366-396	1-3,7,9,20
Y	CCITT, Volume VIII, DATA COMMUNICATION NETWORKS DIRECTORY RECOMENDATIONS X.500-X.521 ISBN 92-61-03731-3	1-3,7,9,20

[X] Further documents are listed in the continuation of Box C

[ ] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 December 1995	18 DEC 1995
Name and mailing address of the ISA/AU AUSTRALIAN INDUSTRIAL PROPERTY ORGANISATION PO BOX 200 WODEN ACT 2606 AUSTRALIA Facsimile No.: (06) 285 3929	Authorized officer **S. LEE** Telephone No.: (06) 283 2205

Form PCT/ISA/210 (second sheet) (July 1992) copmak

INTERNATIONAL SEARCH REPORT

International Application No.
PCT/ AU 95/00560

Box I **Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)**

This International Search Report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a)

Box II **Observations where unity of invention is lacking (Continuation of item 2 of first sheet)**

This International Searching Authority found multiple inventions in this international application, as follows:

1. Claims 1-7, 9, 14-19 relate to an X500 database.
2. Claim 8 is a relational database.
3. Claim 10 is a method of searching a database.
4. Claims 11-13 are to another method of searching a database.

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims

2. ☒ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.